Manage Messages in Azure Service Bus and Azure Queue Storage

By **Wayne Hoggett**

Cancel

Project Guide

Manage Messages with Azure Service Bus and Azure Queue Storage

Introduction

In this hands-on lab, you will manage messages using Azure Service Bus and Azure queue storage. You'll create a Service Bus Queue, configure its properties, and send and receive messages via a simple .NET console application built in Cloud Shell. Similarly, you'll create an Azure Storage Queue, and use another .NET application to send and receive messages, leveraging identity-based authentication.

Solution

Log in to the Azure portal using the credentials provided on the lab page. Be sure to use an incognito or private browser window to ensure you're using the lab account rather than your own.

Create an Azure Service Bus Queue

1. Click the Name link–it'll start with **sbns-**–to the right of the **Service Bus Namespace** resource from the list of resources.

2. From the left navigation menu, under **Entities**, select **Queues**.
3. Click **+ Queue**.
4. For the **Name** option type `myqueue`.

   > **Note**: Remember this value as you will use this later.

5. For the **Max delivery option** option, type `1`.
6. For the **Lock duration** option, type `1` into the **Minutes** field. (It may already be entered, in which case just leave it as is.)

   > **Note**: A message will only be delivered once. If it is not processed within one minute, it will be moved to the dead letter queue. When a message is received from this queue, it will be locked and hidden from other receivers for one minute.

7. Click **Create**.
8. From the left menu, under **Settings**, select **Shared access policies**.
9. From the list of Policies, click the **RootManageSharedAccessKey** link.
10. Click the **Copy to clipboard** icon at the end of the **Primary connection string** field, and store it locally for later use.

Work with Messages in an Azure Service Bus Queue

1. Open **Cloud Shell** by clicking the icon in the top menu.
2. Click **Bash**.
3. Ensure **No storage account required** is selected.
4. For **Subscription**, select the existing subscription.
5. Click **Apply**.
6. In the Cloud Shell menu, click **Settings** and select **Go to Classic version**.
7. Create a new .Net Console application by running the following command:

   ```
   dotnet new console -o servicebus
   ```
8. Change into the project directory, and open Visual Studio Code by running the following command in Cloud Shell:

   ```
   cd servicebus
   ```
9. Open Visual Studio Code above the prompt by running the following command in Cloud Shell:

```
code .
```

> **Note**: You can maximize the Cloud Shell window to make it easier to work with Visual Studio Code from within Cloud Shell.

10. Add the required packages to work with Service Bus by running the following command (still at the `cloud` command prompt, like for the previous commands):

    ```
    dotnet add package Azure.Messaging.ServiceBus
    ```
11. Open **Program.cs** from the left menu in Visual Studio Code.
12. Replace content starter code in **Program.cs** with the code from GitHub.
13. Replace the placeholder values `<CONNECTION-STRING>` and `<QUEUE-NAME>` in **Program.cs** with the values that you copied earlier.
14. Save **Program.cs** by pressing `Control+S` on Windows or Linux or `Command+S` on Mac.

    Optionally review the code.
15. Run the program by running the following command:

    ```
    dotnet run
    ```
16. Use the options in the program to send and receive messages from the Service Bus queue: enter `1`
17. Enter `Test message`
18. Enter `2` to receive the message from the queue.

    `Received message: Test message` will be returned.
19. When you are finished testing the application, type `3` (and press `enter`) to exit the program.

Create an Azure Storage Account Queue

1. Minimize Cloud Shell.
2. In the breadcrumbs at the top of the page, click the Resource Group name. It will have **manage-messages-in-azure** in it.
3. Click the **Name** link of the **Storage account** resource from the list of resources.
4. From the left menu, under **Data storage**, select **Queues**.
5. Click **+ Queue**.
6. Name the queue `myqueue`.

   Remember this name as you will use it later.

7. Click **OK**
8. Copy the Storage account name, which likely starts with **st**, towards the upper-left of the window, and store it locally. You will need it later in the lab.

Work with Messages in an Azure Storage Account Queue

1. Re-open **Cloud Shell** by selecting the icon in the top menu.
2. Restart Cloud Shell by clicking the **Restart Cloud Shell** icon at the top. It looks like a power button.
3. Click **Restart**.
4. Create a new .Net Console application by running the following command:

   ```
   dotnet new console -o storagequeues
   ```
5. Change into the project directory by running the following command:

   ```
   cd storagequeues
   ```
6. Open Visual Studio Code by running the following command:

   ```
   code .
   ```
7. Add the required packages to work with Azure storage queues and the Microsoft identity platform by running the following commands:

   ```
   dotnet add package Azure.Storage.Queues
   dotnet add package Azure.Identity
   ```

   > **Note**: More details on Azure Identity library are available here: Azure Identity client library for .NET

8. Open **Program.cs** from the left menu in Visual Studio Code.
9. Replace content starter code in **Program.cs** with the code from GitHub.
10. Replace the placeholder values `<STORAGE-ACCOUNT-NAME>` and `<QUEUE-NAME>` in **Program.cs**.
11. Save **Program.cs** by pressing `Control+S` on Windows or Linux or `Command+S` on Mac.
12. Optionally review the code.

    > **Note**: This code is using `DefaultAzureCredential` which will use your Azure CLI (Cloud Shell) credentials, later when this application is deployed, it can use a managed identity for passwordless authentication.

13. Run the program by running the following command:

    ```
    dotnet run
    ```

14. Use the options in the program to send and receive messages from the Azure storage queue: enter `1`
15. Enter `Test message`
16. Enter `2` to receive the message from the queue.

    `Received message: Test message` will be returned.

17. When finished, type `3` and (press `enter`) to exit the program.

Conclusion

Congratulations—you've completed this hands-on lab!

Hang tight while we start your Hands-on Lab. Your environment will be ready in approximately 00:45

Creating Lab Environment

Lab Tools

Lab Diagram

Learning Objectives

Successfully complete this lab by achieving the following learning objectives.

1. Create an Azure Service Bus Queue
2. Work with Messages in an Azure Service Bus Queue
3. Create an Azure Storage Account Queue

4 Work with Messages in an Azure Storage Account Queue